

1. Consider each of the following systems from each of three perspectives -

- (i) As a computational problem to be solved by Gaussian Elimination in a three digit, rounding machine (you may wish to check your answers with the laboratory routine `ge_steps_chop.m`)
- (ii) As a geometric problem involving the intersection of two lines
- (iii) As a geometric problem involving construction of one (column) vector in terms of two others.

Which of these would you expect to be ill-conditioned and which to be well conditioned? What attributes appear to be associated with well-conditioned problems from each geometric view?

$$\begin{array}{rclcl} \text{(a.)} & 2.01 x_1 & - & 1.99 x_2 & = & 4.00 \\ & 1.99 x_1 & + & 2.01 x_2 & = & 4.00 \end{array}$$

solution:

a.(i) The augmented matrix here is:

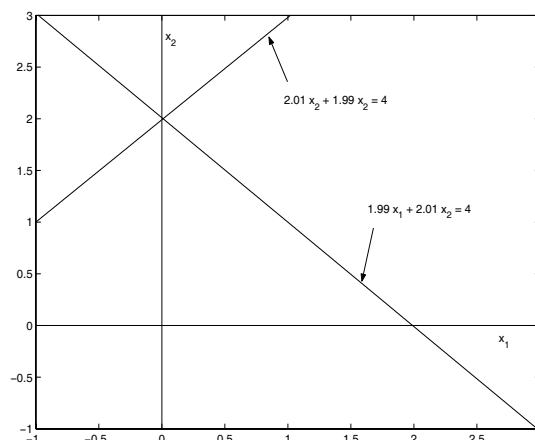
$$\begin{bmatrix} 2.01 & -1.99 & 4.00 \\ 1.99 & 2.01 & 4.00 \end{bmatrix}$$

Elimination produces:

$$R_2 - (.990)R_1 \quad \begin{bmatrix} 2.01 & -1.99 & 4.00 \\ 0 & 3.89 & 0.0400 \end{bmatrix}$$

There are no small pivots, so this system should be relatively well-conditioned.

(ii) Geometrically, solving this system is equivalent to finding the point of intersection of the two straight lines:



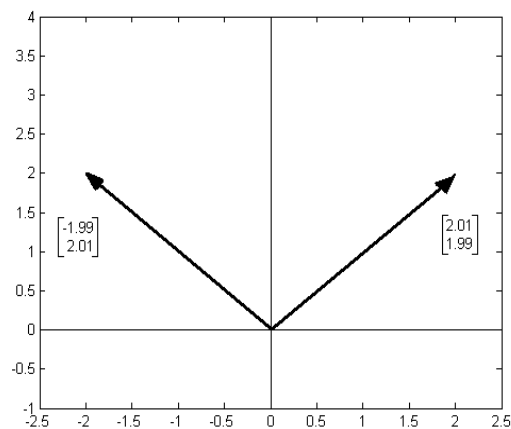
solution:

a.(ii) (cont) Since small perturbations of these lines would not seem move that intersection significantly, from this perspective, we also expect this system to be well-conditioned.

(iii) This system is also equivalent to the problem:

$$x_1 \begin{bmatrix} 2.01 \\ 1.99 \end{bmatrix} + x_2 \begin{bmatrix} -1.99 \\ 2.01 \end{bmatrix} = \begin{bmatrix} 4.00 \\ 4.00 \end{bmatrix}$$

The two column vectors here can be plotted in \mathbb{R}^2 and are



$$\begin{aligned} \text{(b.)} \quad & 2.01 x_1 + 1.99 x_2 = 4.00 \\ & 1.99 x_1 + 2.01 x_2 = 4.00 \end{aligned}$$

solution:

b.(i) The augmented matrix here is:

$$\begin{bmatrix} 2.01 & 1.99 & 4.00 \\ 1.99 & 2.01 & 4.00 \end{bmatrix}$$

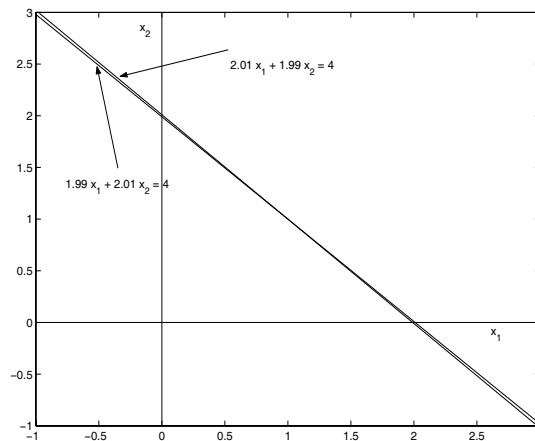
Here, elimination produces:

$$R_2 - (.990)R_1 \quad \begin{bmatrix} 2.01 & 1.99 & 4.00 \\ 0 & 0.0400 & 0.0400 \end{bmatrix}$$

solution:

Here there is an inescapable small pivot, so this system is likely somewhat ill-conditioned. (That can be true even though in this case, Gaussian elimination produces a very accurate solution. To better understand this last point, observe what happens to the solution if you replace the right-hand side by $[4.02 \ 3.99]^T$.)

b. (ii) Geometrically, solving this system is equivalent to finding the point of intersection of the two straight lines:



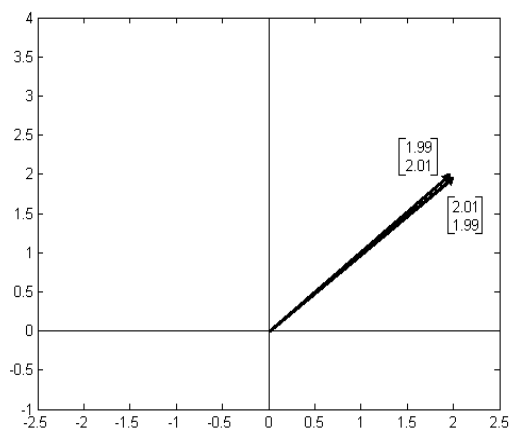
Since small perturbations of these lines would seem move that intersection significantly, from this perspective, from this perspective we also expect this system to be ill-conditioned.

(iii) This system is also equivalent to the problem:

$$x_1 \begin{bmatrix} 2.01 \\ 1.99 \end{bmatrix} + x_2 \begin{bmatrix} 1.99 \\ 2.01 \end{bmatrix} = \begin{bmatrix} 4.00 \\ 4.00 \end{bmatrix}$$

solution:

The two column vectors here can be plotted in \mathbb{R}^2 and are



These vectors are nearly parallel, and drawing a parallelogram with them as sides would be quite dicey.

$$\begin{aligned} \text{(c.)} \quad & 1.01 x_1 + 0.01 x_2 = 4.00 \\ & 0.01 x_1 + 1.01 x_2 = 2.00 \end{aligned}$$

solution:

c.(i) Here, the augmented matrix is:

$$\begin{bmatrix} 1.01 & 0.01 & 4.00 \\ 0.01 & 1.01 & 2.00 \end{bmatrix}$$

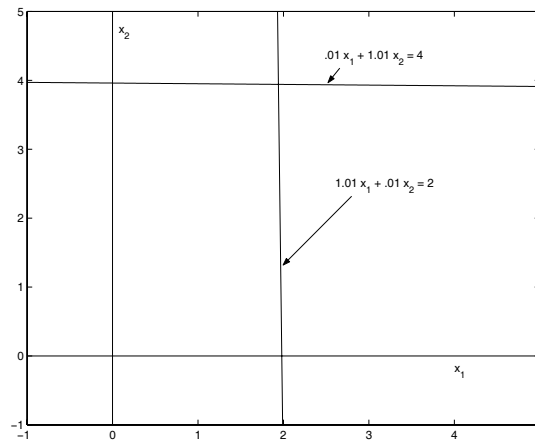
Elimination produces: and then eliminate in the first column:

$$R_2 - (.00990)R_1 \quad \begin{bmatrix} 1.01 & 0.01 & 4.00 \\ 0 & 1.01 & 1.96 \end{bmatrix}$$

The lack of small pivots here suggests that this system is probably fairly well-conditioned.

solution:

c. (ii) Geometrically, solving this system is equivalent to finding the point of intersection of the two straight lines:

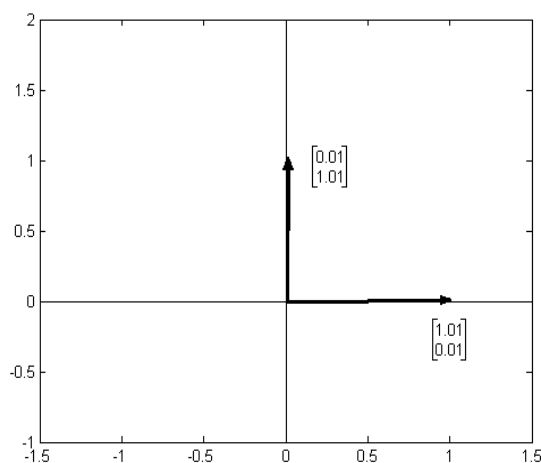


Since small perturbations of these lines would not seem move that intersection significantly, from this perspective, we also expect this system to be well-conditioned.

(iii) This system is also equivalent to the problem:

$$x_1 \begin{bmatrix} 0.01 \\ 1.01 \end{bmatrix} + x_2 \begin{bmatrix} 1.01 \\ 0.01 \end{bmatrix} = \begin{bmatrix} 2.00 \\ 4.00 \end{bmatrix}$$

The two column vectors here can be plotted in \mathbb{R}^2 and are



solution:

There is nothing in this geometry to suggest that constructing the vector

$$\begin{bmatrix} 2.00 & 4.00 \end{bmatrix}^T$$

in terms of these would be difficult.

2. Using MATLAB to calculate the inverses in each case, determine the condition number (using the infinity norm) for each of the matrices in problems 1. Do these values support your geometric intuition about which problems are and are not well-conditioned?

solution:

For problem 1 a.:

$$\mathbf{A} = \begin{bmatrix} 2.01 & -1.99 \\ 1.99 & 2.01 \end{bmatrix} \quad \text{and} \quad \mathbf{A}^{-1} = \begin{bmatrix} 0.251 & 0.249 \\ -0.249 & 0.251 \end{bmatrix}$$

$$\text{and so } \|\mathbf{A}\|_{\infty} \|\mathbf{A}^{-1}\|_{\infty} = (4)(0.5) = 2.$$

For problem 1 b.:

$$\mathbf{A} = \begin{bmatrix} 2.01 & 1.99 \\ 1.99 & 2.01 \end{bmatrix} \quad \text{and} \quad \mathbf{A}^{-1} = \begin{bmatrix} 25.1 & -24.9 \\ -24.9 & 25.1 \end{bmatrix}$$

$$\text{and so } \|\mathbf{A}\|_{\infty} \|\mathbf{A}^{-1}\|_{\infty} = (4)(50) = 200.$$

For problem 1 c.:

$$\mathbf{A} = \begin{bmatrix} 0.01 & 1.01 \\ 1.01 & 0.01 \end{bmatrix} \quad \text{and} \quad \mathbf{A}^{-1} = \begin{bmatrix} -0.00980 & 0.990 \\ 0.990 & -0.00980 \end{bmatrix}$$

$$\text{and so } \|\mathbf{A}\|_{\infty} \|\mathbf{A}^{-1}\|_{\infty} = (1.02)(1.00) = 1.02.$$

So, in summary

<u>Problem Number</u>	<u>$\ \mathbf{A}\ _{\infty} \ \mathbf{A}^{-1}\ _{\infty}$</u>	<u>Apparent Condition</u>
1 a.	2	good
1 b.	200	not good
1 c.	1.02	good

These data strongly supports the notion that “bad” problems are those with “large(r)” condition numbers.

3. Based on your results from problems 1 and 2, what geometric attributes would you associate with the best-conditioned systems of two linear equations in two unknowns?

solution:

It looks a if

- (i) There should be no small pivots in the echelon form of the augmented matrix.
- (ii) The lines represented by the equations should not be nearly parallel.
- (ii) The vectors representing the columns of the original augmented matrix should not be nearly parallel. (In fact, likely the closer they are to orthogonal, the better!)

4. Use MATLAB to generate random 6×6 matrices with condition numbers satisfying:

- a. $1 < \text{cond}(\mathbf{A}) \leq 10$
- b. $10 < \text{cond}(\mathbf{A}) \leq 100$
- c. $100 < \text{cond}(\mathbf{A}) \leq 1000$
- d. $1000 < \text{cond}(\mathbf{A})$

For each of the matrices, have MATLAB compute \mathbf{A}^{-1} , and observe the correlations, if any, between $\text{cond}(\mathbf{A})$, the elements of \mathbf{A} , and the elements of \mathbf{A}^{-1} .

solution:

The matrix for part a. can be generated with the MATLAB commands:

`a = rand(6); while (cond(a) > 10) ; a = rand(6) ;end`

Since the matrix will be (quasi-)random, the results will differ depending on the machine used, and what, if any computations had been done on that machine before. On the machine used to create these solutions, the matrix obtained was:

$$\mathbf{a} = \begin{bmatrix} 0.7848 & 0.2109 & 0.1770 & 0.4418 & 0.0418 & 0.4364 \\ 0.2409 & 0.5162 & 0.7494 & 0.1039 & 0.0570 & 0.8193 \\ 0.8223 & 0.1472 & 0.6682 & 0.8750 & 0.0590 & 0.4448 \\ 0.0126 & 0.8199 & 0.2478 & 0.8925 & 0.4735 & 0.9153 \\ 0.5697 & 0.3899 & 0.8984 & 0.0169 & 0.3840 & 0.4911 \\ 0.4619 & 0.0121 & 0.2786 & 0.5165 & 0.5044 & 0.7484 \end{bmatrix}$$

According to MATLAB, the condition number of this matrix is 9.4478, and

$$\mathbf{a}^{-1} = \begin{bmatrix} 1.5524 & -0.4806 & -0.3734 & -0.2627 & 0.4951 & -0.1608 \\ 1.0492 & -0.4357 & -0.6065 & 0.9738 & 0.8017 & -1.4912 \\ -1.4127 & 0.3506 & 1.0148 & -0.2571 & 0.4623 & -0.1520 \\ -0.6759 & -0.3876 & 1.1337 & 0.4458 & -0.4840 & -0.0828 \\ -0.4018 & -1.5294 & -0.5628 & 0.4833 & 1.4158 & 0.7229 \\ 0.2883 & 1.4713 & -0.5408 & -0.3913 & -1.1106 & 1.0860 \end{bmatrix}$$

and the largest (magnitude) element is the one in the (1,1) position, i.e. 1.5524.

solution:

On our system, the matrix for part b. was generated by the MATLAB command:

while (cond(a) < 10) + (cond(a) > 100) ; a = rand(6) ; end
and was:

$$\mathbf{a} = \begin{bmatrix} 0.3421 & 0.2422 & 0.5652 & 0.9219 & 0.7380 & 0.9972 \\ 0.5281 & 0.0367 & 0.8852 & 0.7104 & 0.2497 & 0.0913 \\ 0.0260 & 0.0185 & 0.3396 & 0.1444 & 0.6538 & 0.4329 \\ 0.2106 & 0.1977 & 0.2250 & 0.9793 & 0.6498 & 0.7756 \\ 0.5196 & 0.2106 & 0.5690 & 0.8176 & 0.8832 & 0.8888 \\ 0.8871 & 0.9244 & 0.7802 & 0.2841 & 0.5109 & 0.3699 \end{bmatrix}$$

whose condition number was 23.1984. MATLAB also provided

$$\mathbf{a}^{-1} = \begin{bmatrix} -1.6672 & -0.3255 & -2.3206 & -1.4592 & 4.3872 & -0.1915 \\ 0.4340 & -0.2851 & 0.8146 & 1.7056 & -2.8634 & 1.2508 \\ 2.0961 & 0.8936 & 1.0879 & -1.0832 & -2.0682 & 0.0965 \\ -0.7588 & 0.6561 & -0.1816 & 2.0431 & -0.8862 & -0.0585 \\ -2.9759 & 0.0571 & 1.9113 & 1.7800 & 0.7728 & 0.1823 \\ 3.1851 & -0.9744 & -1.2650 & -2.5059 & 0.6102 & -0.3735 \end{bmatrix}$$

whose maximum magnitude element ((1,5) position) is 4.3872.

On our system, the matrix for part c., created by the MATLAB commands:

while (cond(a) < 100) + (cond(a) > 1000) ; a = rand(6) ; end
was

$$\mathbf{a} = \begin{bmatrix} 0.9544 & 0.2858 & 0.7447 & 0.9998 & 0.1917 & 0.2744 \\ 0.5390 & 0.8735 & 0.3678 & 0.5454 & 0.6127 & 0.1711 \\ 0.9245 & 0.7432 & 0.4518 & 0.8581 & 0.0167 & 0.3546 \\ 0.4856 & 0.7043 & 0.3724 & 0.1571 & 0.8871 & 0.5946 \\ 0.4054 & 0.6899 & 0.2006 & 0.7204 & 0.1290 & 0.1387 \\ 0.3072 & 0.1251 & 0.5697 & 0.7503 & 0.6581 & 0.3183 \end{bmatrix}$$

and had a condition number of about 557.6.

solution:

MATLAB also produced for \mathbf{a}^{-1} :

$$\begin{bmatrix} 34.0429 & -18.2592 & -42.4159 & 19.3378 & 39.6221 & -25.6732 \\ -21.8059 & 12.9070 & 27.7139 & -12.3899 & -25.8580 & 15.4015 \\ -57.4484 & 34.2937 & 75.0372 & -34.2344 & -73.3817 & 43.4317 \\ 14.5612 & -9.3103 & -19.3486 & 8.2452 & 20.0168 & -10.1201 \\ 24.4064 & -13.0046 & -32.2660 & 14.2690 & 29.7618 & -17.7312 \\ -6.2481 & 0.0045 & 8.0617 & -1.4581 & -5.4559 & 4.6463 \end{bmatrix}$$

whose largest magnitude element ((3,3) position) is 75.0372.

Finally, on our system, the matrix for part d., created by the MATLAB commands:

while (cond(a) < 1000) ; a = rand(6) ; end

was

$$\mathbf{a} = \begin{bmatrix} 0.8755 & 0.3270 & 0.3715 & 0.2942 & 0.3295 & 0.7543 \\ 0.1990 & 0.1833 & 0.8191 & 0.4704 & 0.9536 & 0.5051 \\ 0.3554 & 0.7504 & 0.3740 & 0.6137 & 0.5733 & 0.6746 \\ 0.4652 & 0.7498 & 0.4528 & 0.9855 & 0.3979 & 0.1903 \\ 0.3119 & 0.2033 & 0.3784 & 0.4298 & 0.1470 & 0.0977 \\ 0.0856 & 0.7711 & 0.5556 & 0.8799 & 0.2781 & 0.2451 \end{bmatrix}$$

whose condition number was approximately 10,709, and whose inverse, according to MATLAB, is:

$$\begin{bmatrix} 192.95 & 81.61 & -314.44 & 159.07 & -520.81 & 187.40 \\ 557.81 & 236.84 & -910.32 & 458.09 & -1513.02 & 547.77 \\ 316.52 & 134.85 & -517.11 & 257.68 & -855.09 & 311.81 \\ -624.88 & -265.78 & 1019.69 & -511.94 & 1693.24 & -612.74 \\ 114.92 & 50.19 & -188.33 & 96.75 & -314.89 & 111.55 \\ -426.81 & -182.08 & 698.83 & -352.72 & 1158.68 & -418.28 \end{bmatrix}$$

The largest magnitude element of this matrix ((4,5) position) is 1693.24

solution:

The results here can be summarized in the table

<u>cond(a)</u>	<u>max(a_{ij})</u>	<u>max(a_{ij}^{-1})</u>
9.4478	0.9153	1.5524
23.1984	0.9972	4.3872
557.5909	0.9998	75.0372
10709.	0.9855	1693.24

Notice that the maximum element in **a** is always less than one in magnitude. Therefore, in this case, we expect that since:

$$\kappa(\mathbf{a}) = \|\mathbf{a}\| \|\mathbf{a}^{-1}\|$$

then, in this case, we clearly expect poorly conditioned matrices to correspond to those with “large” inverses. The correspondence of the largest elements of \mathbf{a}^{-1} with the condition numbers here supports that conjecture.

5. Consider the two systems

$$\begin{array}{rclclcl} 4.55 x_1 & + & 2.39 x_2 & + & 4.18 x_3 & = & 7.52 \\ 2.40 x_1 & + & 2.04 x_2 & + & 3.75 x_3 & = & 4.77 \\ 3.19 x_1 & - & 2.06 x_2 & - & 4.23 x_3 & = & 1.45 \end{array}$$

and

$$\begin{array}{rclclcl} 4.55 x_1 & + & 2.39 x_2 & + & 4.18 x_3 & = & 7.56 \\ 2.40 x_1 & + & 2.04 x_2 & + & 3.75 x_3 & = & 4.72 \\ 3.19 x_1 & - & 2.06 x_2 & - & 4.23 x_3 & = & 1.50 \end{array}$$

(Notice the left-hand sides here are identical, and the right-hand side of the second is only a “relatively small” perturbation of the right-hand side of the first.)

a. Compare the exact (MATLAB) solutions to both problems. What does that comparison suggest about the condition of this matrix?

solution:

Using MATLAB, the calculated solution to the first system is

$$\begin{bmatrix} 1.125590 \dots \\ 0.799300 \dots \\ 0.116802 \dots \end{bmatrix}$$

while the solution to the second one is:

$$\mathbf{x} = \begin{bmatrix} 1.023110 \dots \\ 3.279168 \dots \\ -1.179991 \dots \end{bmatrix}$$

Observe these solutions are drastically different, especially in the second and third components, even though they came from the same basic system with only slightly changed right-hand sides. This is classically characteristic behavior for so-called *ill-conditioned* systems.

b. Using MATLAB, calculate the condition number of the matrix in this problem. Does your result support your conclusion in part a or not?

solution:

Using MATLAB, the calculated condition number of the matrix is

$$\kappa(\mathbf{A}) \doteq 584$$

This implies that relative errors or changes in the data may produce changes in the solution of up to almost six hundred times greater. This is a relatively large condition number, and we should have expected, had we known this, that the resulting matrix will be at least somewhat ill-conditioned, and that small changes in the right-hand side data may produce large swings in the solution. In fact, in this instance, the two right-hand sides:

$$\mathbf{b}^{(1)} = \begin{bmatrix} 7.52 \\ 4.77 \\ 1.45 \end{bmatrix} \quad \text{and} \quad \mathbf{b}^{(2)} = \begin{bmatrix} 7.56 \\ 4.72 \\ 1.50 \end{bmatrix}$$

lead to, using MATLAB's **norm()** command

$$\frac{\|\mathbf{b}^{(1)} - \mathbf{b}^{(2)}\|}{\|\mathbf{b}^{(1)}\|} = .0090$$

and so, in the worst case, we could have expected to see here

$$\frac{\|\mathbf{x}^{(1)} - \mathbf{x}^{(2)}\|}{\|\mathbf{x}^{(1)}\|} \doteq 584(.0090) = 5.25$$

or values that change by over five hundred percent. In this case

$$\frac{\|\mathbf{x}^{(1)} - \mathbf{x}^{(2)}\|}{\|\mathbf{x}^{(1)}\|} \doteq 2.02$$

so we actually did better than worst case. However, the solutions are clearly nowhere close to each other, even though the “data” are.

6. Consider the system of equations:

$$x_1 + \frac{1}{2} x_2 + \frac{1}{3} x_3 + \frac{1}{4} x_4 = \frac{4}{3}$$

$$\frac{1}{2} x_1 + \frac{1}{3} x_2 + \frac{1}{4} x_3 + \frac{1}{5} x_4 = \frac{5}{6}$$

$$\frac{1}{3} x_1 + \frac{1}{4} x_2 + \frac{1}{5} x_3 + \frac{1}{6} x_4 = \frac{19}{30}$$

$$\frac{1}{4} x_1 + \frac{1}{5} x_2 + \frac{1}{6} x_3 + \frac{1}{7} x_4 = \frac{31}{60}$$

(The matrix here is an example of the so-called *Hilbert Matrix*, and is classic in numerical analysis.)

(a.) Simulate the solution of this system using Gaussian elimination on a three-digit, decimal based computer which rounds all calculations, including intermediate ones. (Note the true solution here is $\mathbf{x} = [1 \quad -2 \quad 4 \quad 0]^T$.)

solution:

The augmented matrix is, in three-digit arithmetic:

$$\begin{bmatrix} 1.00 & 0.500 & 0.333 & 0.250 & 1.33 \\ 0.500 & 0.333 & 0.250 & 0.200 & 0.833 \\ 0.333 & 0.250 & 0.200 & 0.167 & 0.633 \\ 0.250 & 0.200 & 0.167 & 0.143 & 0.517 \end{bmatrix}$$

Eliminate the first column:

$$\begin{array}{l} R_2 - (0.500)R_1 \\ R_3 - (0.333)R_1 \\ R_4 - (0.250)R_1 \end{array} \begin{bmatrix} 1.00 & 0.500 & 0.333 & 0.250 & 1.33 \\ 0 & 0.0830 & 0.0830 & 0.0750 & 0.168 \\ 0 & 0.0830 & 0.0890 & 0.0837 & 0.190 \\ 0 & 0.0750 & 0.0837 & 0.0805 & 0.184 \end{bmatrix}$$

Continue to eliminate the second column:

$$\begin{array}{l} R_3 - R_2 \\ R_4 - (0.904)R_2 \end{array} \begin{bmatrix} 1.00 & 0.500 & 0.333 & 0.250 & 1.33 \\ 0 & 0.0830 & 0.0830 & 0.0750 & 0.168 \\ 0 & 0 & 0.00600 & 0.00870 & 0.0220 \\ 0 & 0 & 0.00870 & 0.0127 & 0.0320 \end{bmatrix}$$

solution:

and then the third column:

$$R_4 - (1.45)R_3 \quad \begin{bmatrix} 1.00 & 0.500 & 0.333 & 0.250 & 1.33 \\ 0 & 0.0830 & 0.0830 & 0.0750 & 0.168 \\ 0 & 0 & 0.00600 & 0.00870 & 0.0220 \\ 0 & 0 & 0 & 0.000100 & 0.000100 \end{bmatrix}$$

Note the appearance of an inescapable and extremely small pivot on the diagonal in the echelon form.

Back substitute:

$$x_4 = \frac{.0001}{.0001} = 1.00$$

$$\begin{aligned} x_3 &= \frac{.0220 - .0087x_4}{.006} = \frac{.0220 - \overbrace{(.0087)(1.00)}^{0.0087}}{.006} \\ &= \frac{.0220 - 0.0087}{.006} = \frac{0.0133}{.006} = 2.22 \end{aligned}$$

$$\begin{aligned} x_2 &= \frac{.168 - .0830x_3 - .0750x_4}{.0830} = \frac{.168 - \overbrace{(.0830)(2.22)}^{0.18426} - \overbrace{(.0750)(1.00)}^{0.0750}}{.0830} \\ &= \frac{.168 - .184 - .075}{.0830} = \frac{-.091}{.0830} = -1.10 \end{aligned}$$

$$\begin{aligned} x_1 &= \frac{1.33 - .500x_2 - .333x_3 - .250x_4}{1.00} \\ &= 1.33 - \overbrace{(.500)(-1.10)}^{-.550} - \overbrace{(.333)(2.22)}^{.73926} - \overbrace{(.250)(1.00)}^{.250} \\ &= 1.33 + .550 - .739 - .250 = .891 \end{aligned}$$

(b.) For your computed solution to part a. above, determine the error (**e**), the residual (**r**).

solution:

The true solution to this problem is $[1 \ -2 \ 4 \ 0]^T$, and therefore the error is

$$\mathbf{x} - \tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ -2 \\ 4 \\ 0 \end{bmatrix} - \begin{bmatrix} 0.891 \\ -1.10 \\ 2.22 \\ 1.00 \end{bmatrix} = \begin{bmatrix} 0.109 \\ -0.900 \\ 1.78 \\ 1.00 \end{bmatrix}$$

Note the solution is awful!

Using the three-digit version of the matrix and right-hand side, the residual corresponding to this solution is, using MATLAB to ensure an accurate result:

$$\mathbf{r} = \begin{bmatrix} 1.33 \\ 0.833 \\ 0.633 \\ 0.517 \end{bmatrix} - \begin{bmatrix} 1.00 & 0.500 & 0.333 & 0.250 \\ 0.500 & 0.333 & 0.250 & 0.200 \\ 0.333 & 0.250 & 0.200 & 0.167 \\ 0.250 & 0.200 & 0.167 & 0.143 \end{bmatrix} \begin{bmatrix} 0.891 \\ -1.10 \\ 2.22 \\ 1.00 \end{bmatrix} = \begin{bmatrix} -0.000260 \\ -0.001200 \\ 0.000297 \\ 0.000510 \end{bmatrix}$$

Note the residual appear acceptably small relative to the original right-hand side. Yet the solution is awful! Small residuals may mean nothing!

(c.) Using MATLAB to invert this matrix, determine its condition number in the infinity norm.

solution:

For the original matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix} \implies \|\mathbf{A}\|_{\infty} = 2\frac{1}{6} \doteq 2.083$$

(sum the first row).

solution:

However, using MATLAB, we find (to three digits):

$$\mathbf{A}^{-1} = \begin{bmatrix} 16.0 & -120. & 240. & -140. \\ -120. & 1200. & -2700. & 1680. \\ 240. & -2700. & 6480. & -4200. \\ -140. & 1680. & -4200. & 2800. \end{bmatrix} \Rightarrow \|\mathbf{A}^{-1}\|_{\infty} = 13620$$

and therefore the condition number of this matrix, in the infinity norm, is approximately 28,400.

(d.) Do your answers to part (b.) seem consistent with the condition number you found in part (c.)?

solution:

The condition number found above is very large, i.e. this is a very ill-conditioned matrix. So we should expect both poor numerical solutions, and small residuals to necessarily signify anything.

7. a. Show that for any matrices \mathbf{A} and \mathbf{B} in $\mathbb{C}^{m \times m}$, and any scalar α ,

$$\kappa(\alpha \mathbf{A}) = \kappa(\mathbf{A}) \quad , \quad \kappa(\mathbf{A}^{-1}) = \kappa(\mathbf{A}) \quad \text{and} \quad \kappa(\mathbf{A} \mathbf{B}) \leq \kappa(\mathbf{A}) \kappa(\mathbf{B})$$

solution:

By definition, in any given norm,

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$$

But, we know that norms are linear. Therefore, for any scalar α ,

$$\|\alpha \mathbf{A}\| = |\alpha| \|\mathbf{A}\|$$

But also, by the properties of inverses

$$(\alpha \mathbf{A})^{-1} = \mathbf{A}^{-1} \alpha^{-1} = \frac{1}{\alpha} \mathbf{A}^{-1} \implies \|(\alpha \mathbf{A})^{-1}\| = \frac{1}{|\alpha|} \|\mathbf{A}^{-1}\|$$

Therefore, by definition

$$\begin{aligned} \kappa(\alpha \mathbf{A}) &= \|\alpha \mathbf{A}\| \cdot \|(\alpha \mathbf{A})^{-1}\| = (|\alpha| \|\mathbf{A}\|) \left(\frac{1}{|\alpha|} \|\mathbf{A}^{-1}\| \right) \\ &= \|\mathbf{A}\| \|\mathbf{A}^{-1}\| = \kappa(\mathbf{A}) \end{aligned}$$

and so this proves the first part.

We also know that, for any norm, and any \mathbf{A} and \mathbf{B} ,

$$\|\mathbf{A} \mathbf{B}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|$$

and also, since $(\mathbf{A} \mathbf{B})^{-1} = \mathbf{B}^{-1} \mathbf{A}^{-1}$, then

$$\|(\mathbf{A} \mathbf{B})^{-1}\| = \|\mathbf{B}^{-1} \mathbf{A}^{-1}\| \leq \|\mathbf{B}^{-1}\| \cdot \|\mathbf{A}^{-1}\|$$

Putting these together with the definition of condition number yields

$$\begin{aligned} \kappa(\mathbf{A} \mathbf{B}) &= \|\mathbf{A} \mathbf{B}\| \cdot \|(\mathbf{A} \mathbf{B})^{-1}\| \\ &\leq (\|\mathbf{A}\| \cdot \|\mathbf{B}\|) (\|\mathbf{B}^{-1}\| \cdot \|\mathbf{A}^{-1}\|) \\ &= \underbrace{\|\mathbf{A}\| \|\mathbf{A}^{-1}\|}_{\kappa(\mathbf{A})} \cdot \underbrace{\|\mathbf{B}\| \|\mathbf{B}^{-1}\|}_{\kappa(\mathbf{B})} \end{aligned}$$

solution:

This last inequality proves the second result. (We would add that, in practice, the condition number of the product is generally at least one order of magnitude smaller than the product of the individual condition numbers, however.)

Finally, by definition

$$\begin{aligned}\kappa(\mathbf{A}^{-1}) &= \|\mathbf{A}^{-1}\| \cdot \|(\mathbf{A}^{-1})^{-1}\| = \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\| \\ &= \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| = \kappa(\mathbf{A})\end{aligned}$$

which proves the final part.

b. Using the singular value decomposition, show that, for any $\mathbf{A} \in \mathbb{C}^{m \times n}$,

$$\kappa(\mathbf{Q}\mathbf{A}) = \kappa(\mathbf{A})$$

whenever the columns of \mathbf{Q} are orthonormal, i.e. whenever $\mathbf{Q}^H \mathbf{Q} = \mathbf{I}$, and the condition number is computed in the Euclidean norm.

solution:

For this, we simply need to show that \mathbf{A} and $\mathbf{Q}\mathbf{A}$ have the same singular values. But, the full SVD of \mathbf{A} is

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H$$

where \mathbf{U} and \mathbf{V} are unitary. But then

$$\mathbf{Q}\mathbf{A} = \mathbf{Q}(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^H) = (\mathbf{Q}\mathbf{U})\mathbf{\Sigma}\mathbf{V}^H$$

However, it is easily seen that $(\mathbf{Q}\mathbf{U})^H(\mathbf{Q}\mathbf{U}) = (\mathbf{U}^H\mathbf{Q}^H)(\mathbf{Q}\mathbf{U}) = \mathbf{I}$

Therefore the equation immediately above this last line must represent a (reduced) SVD of $\mathbf{Q}\mathbf{A}$. But since the reduced SVD is unique except for unit scalar multiples of the singular vectors, then the singular values of $\mathbf{Q}\mathbf{A}$ must be precisely the diagonal elements of $\mathbf{\Sigma}$, i.e. the singular values of \mathbf{A} .

c. Why does the first of the results in part a. imply that $\det \mathbf{A}$ “close to” zero is a very *poor* test for nearly singular.

solution:

A fundamental property of determinants is that

$$\det(\alpha \mathbf{A}) = \alpha^m \det(\mathbf{A})$$

But this implies that I can make $\det(\alpha \mathbf{A})$ arbitrary large or small, simply by choosing an appropriate value of α , *without changing the condition number of \mathbf{A}* . Since we have chosen to use the terms nearly singular matrix and ill-conditioned as synonyms, then the value of the determinant bears no relation to the condition number, and hence is useless for determining whether or not a matrix is close to singular.

d. Why do the second and third of these results imply that, in the Euclidean norm, the condition of the normal equations formulation for solving the least squares problem, i.e.:

$$\mathbf{x} = (\mathbf{A}^H \mathbf{A})^{-1} \mathbf{A}^H \mathbf{b}$$

may be as large as $\kappa(\mathbf{A})^3$, while the condition of solving the same problem by **QR** factorization, i.e.:

$$\mathbf{x} = \mathbf{R}^{-1} (\mathbf{Q}^H \mathbf{b})$$

has condition exactly equal to $\kappa(\mathbf{A})$.

solution:

We already know that, in the Euclidean norm, $\kappa(\mathbf{A}^H \mathbf{A}) = \kappa(\mathbf{A})^2$. But then, by the third result above, in any norm,

$$\kappa\left((\mathbf{A}^H \mathbf{A})^{-1}\right) = \kappa(\mathbf{A}^H \mathbf{A}) = \kappa(\mathbf{A})^2$$

But we also know $\kappa(\mathbf{A}^H) = \kappa(\mathbf{A})$, and therefore, by the second result in part a., and the result immediately above

$$\kappa\left((\mathbf{A}^H \mathbf{A})^{-1} \mathbf{A}^H\right) \leq \kappa\left((\mathbf{A}^H \mathbf{A})^{-1}\right) \kappa(\mathbf{A}^H) = \kappa(\mathbf{A})^2 \cdot \kappa(\mathbf{A}) = \kappa(\mathbf{A})^3$$

solution:

Finally, again by the second and third results in part a., plus the result of part b.,

$$\begin{aligned}\kappa(\mathbf{R}^{-1}\mathbf{Q}^H) &= \kappa\left((\mathbf{R}^{-1}\mathbf{Q}^H)^H\right) = \kappa\left(\mathbf{Q}(\mathbf{R}^{-1})^H\right) \\ &= \kappa\left((\mathbf{R}^{-1})^H\right) = \kappa(\mathbf{R}^{-1}) = \kappa(\mathbf{R}) = \kappa(\mathbf{A})\end{aligned}$$

8. Consider the following “problem” from \mathbb{R}^3 to \mathbb{R}^1 :

$$f(\mathbf{x}) = f(x, y, z) = (x + y) - z$$

together with the associated “algorithm”

$$\tilde{f}(\mathbf{x}) = \tilde{f}(x, y, z) = [fl(x) \oplus fl(y)] \ominus fl(z)$$

- a. Simulate the results of this algorithm, for $x = 1$, $y = 0.005444\dots$, and $z = 1$ in rounding, floating point machines using two through six decimal digits, and having an accumulator (ALU) of only the same length.

solution:

For a two-digit machine,

$$fl(x) \oplus fl(y) = 1.0 \times 10^1 \oplus .54 \times 10^{-2} \implies \begin{array}{rcl} & .10 & \times 10^1 \\ + & .00054 & \times 10^1 \\ \hline = & .10 & \times 10^1 \end{array}$$

Therefore:

$$\tilde{f}(\mathbf{x}) = [fl(x) \oplus fl(y)] \ominus fl(z) = [.10 \times 10^1] \ominus .10 \times 10^1 = .00 \times 10^1$$

A similar result holds for a three-digit machine. However, for a four-digit machine:

$$fl(x) \oplus fl(y) = 1.0 \times 10^1 \oplus .5444 \times 10^{-2} \implies \begin{array}{rcl} & .1000 & \times 10^1 \\ + & .0005444 & \times 10^1 \\ \hline = & .1005 & \times 10^1 \end{array}$$

Therefore:

$$[fl(x) \oplus fl(y)] \ominus fl(z) = [.1005 \times 10^1] \ominus .1000 \times 10^1 \implies \begin{array}{rcl} & .1005 & \times 10^1 \\ - & .1000 & \times 10^1 \\ \hline = & .0005 & \times 10^1 \end{array}$$

or, after normalizing

$$\tilde{f}(\mathbf{x}) = [fl(x) \oplus fl(y)] \ominus fl(z) = .5000 \times 10^{-2}$$

Of course, in all cases, the exact result is: $f(\mathbf{x}) = .54444\dots \times 10^{-2}$

b. For each calculation, determine

$$f(\mathbf{x}) \quad , \quad \tilde{f}(\mathbf{x}) \quad , \quad \text{and} \quad \frac{\|\tilde{f}(\mathbf{x}) - f(\mathbf{x})\|}{\|f(\mathbf{x})\|}$$

Discuss whether these results support or fail to support the statements:

- (i.) \tilde{f} is forward stable.
- (ii.) \tilde{f} is backward stable.

solution:

Based on the results in part a., we can develop the following table:

<u>Digits</u>	<u>$f(\mathbf{x})$</u>	<u>$\tilde{f}(\mathbf{x})$</u>	<u>$\frac{\ \tilde{f}(\mathbf{x}) - f(\mathbf{x})\ }{\ f(\mathbf{x})\ }$</u>	<u>$\epsilon_{\text{machine}}$</u>
2	$.54444 \dots \times 10^{-2}$	$.00 \times 10^{-2}$	1.00	$.5 \times 10^{-1}$
3	$.54444 \dots \times 10^{-2}$	$.000 \times 10^{-2}$	1.00	$.5 \times 10^{-2}$
4	$.54444 \dots \times 10^{-2}$	$.5000 \times 10^{-2}$	$.816 \times 10^{-1}$	$.5 \times 10^{-3}$
5	$.54444 \dots \times 10^{-2}$	$.54000 \times 10^{-2}$	$.816 \times 10^{-2}$	$.5 \times 10^{-4}$
6	$.54444 \dots \times 10^{-2}$	$.544000 \times 10^{-2}$	$.816 \times 10^{-3}$	$.5 \times 10^{-5}$
7	$.54444 \dots \times 10^{-2}$	$.5444000 \times 10^{-2}$	$.816 \times 10^{-4}$	$.5 \times 10^{-6}$

For this algorithm to be forward stable (accurate), we must have

$$\frac{\|\tilde{f}(\mathbf{x}) - f(\mathbf{x})\|}{\|f(\mathbf{x})\|} = \mathbf{O}(\epsilon_{\text{machine}})$$

i.e. there must be a constant C , *independent* of x , such that

$$\frac{\|\tilde{f}(\mathbf{x}) - f(\mathbf{x})\|}{\|f(\mathbf{x})\|} \leq C \epsilon_{\text{machine}}$$

Note, in this instance, we could use $C = 200$ and satisfy the above inequality, however, you should convince yourself that if we changed the value of y to $.00054444 \dots$, that value would no longer work. In fact we would have to then use a C closer to 2000. Therefore, it does not appear that C can be selected *independent* of \mathbf{x} , and so this algorithm is **not** forward stable. (That, of course, should have been intuitive, since this is clearly a potential catastrophic cancellation calculation whenever $x = z$.)

solution:

Backward stability requires that, for all \mathbf{x} , we can find an $\tilde{\mathbf{x}}$ such that

$$f(\tilde{\mathbf{x}}) = \tilde{f}(\mathbf{x}) \quad \text{where} \quad \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} = \mathbf{O}(\epsilon_{\text{machine}})$$

But

$$\tilde{f}(\mathbf{x}) = [fl(x) \oplus fl(y)] \ominus fl(z)$$

and we know that

$$fl(x) = x(1 + \epsilon_1) \quad , \quad fl(y) = y(1 + \epsilon_2) \quad , \quad \text{and} \quad fl(z) = z(1 + \epsilon_3) \quad ,$$

where $|\epsilon_i| \leq \epsilon_{\text{machine}}$. Therefore, because of the properties of floating point operations (13.7, p. 99), we also know

$$fl(x) \oplus fl(y) = x(1 + \epsilon_1) \oplus y(1 + \epsilon_2) = [x(1 + \epsilon_1) + y(1 + \epsilon_2)](1 + \epsilon_4)$$

Hence,

$$\begin{aligned} \tilde{f}(\mathbf{x}) &= [fl(x) \oplus fl(y)] \ominus fl(z) \\ &= [x(1 + \epsilon_1) + y(1 + \epsilon_2)](1 + \epsilon_4) \ominus z(1 + \epsilon_3) \\ &= \left[[x(1 + \epsilon_1) + y(1 + \epsilon_2)](1 + \epsilon_4) - z(1 + \epsilon_3) \right](1 + \epsilon_5) \\ &= x(1 + \epsilon_1)(1 + \epsilon_4)(1 + \epsilon_5) + y(1 + \epsilon_2)(1 + \epsilon_4)(1 + \epsilon_5) \\ &\quad - z(1 + \epsilon_3)(1 + \epsilon_5) \end{aligned}$$

Multiplying out the various ϵ_i terms and linearizing yields:

$$\tilde{f}(\mathbf{x}) = x(1 + 3\epsilon_1') + y(1 + 3\epsilon_2') - z(1 + 2\epsilon_3')$$

where, again $|\epsilon_i'| \leq \epsilon_{\text{machine}}$. But, by definition

$$x(1 + 3\epsilon_1') + y(1 + 3\epsilon_2') - z(1 + 2\epsilon_3') = f(\tilde{\mathbf{x}})$$

for $\tilde{\mathbf{x}} = (x(1 + 3\epsilon_1'), y(1 + 3\epsilon_2'), z(1 + 2\epsilon_3'))$.

solution:

But then, by direct computation:

$$\tilde{x} - x = (3\epsilon_1'x, 3\epsilon_2'y, 2\epsilon_3'z) \equiv \underbrace{\begin{bmatrix} 3\epsilon_1' & 0 & 0 \\ 0 & 3\epsilon_2' & 0 \\ 0 & 0 & 2\epsilon_3' \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

But then, by the properties of norms and because \mathbf{A} is diagonal:

$$\begin{aligned} \|\tilde{x} - x\| &= \|\mathbf{A} \mathbf{x}\| \leq \|\mathbf{A}\| \|\mathbf{x}\| \\ &= \max\{3|\epsilon_1'|, 3|\epsilon_2'|, 2|\epsilon_3'|\} \|\mathbf{x}\| \leq 3\epsilon_{\text{machine}} \|\mathbf{x}\| \end{aligned}$$

or, equivalently

$$\tilde{f}(\mathbf{x}) = f(\tilde{\mathbf{x}}) \quad \text{where} \quad \frac{\|\tilde{x} - x\|}{\|\mathbf{x}\|} \leq 3\epsilon_{\text{machine}} = \mathbf{O}(\epsilon_{\text{machine}})$$

Therefore, by definition, the algorithm is backward stable.

9. Consider the vectors

$$\mathbf{x} = [-1.19 \quad -2.20 \quad 0.986]^T \quad \text{and} \quad \mathbf{y} = [-0.519 \quad 0.327 \quad 0.234]^T$$

a. Compute $\mathbf{x}\mathbf{y}^T$ as an exact (infinite precision) result.

solution:

Since the original data is only given to three digits, the exact answer can be computed in any six (or more) digit machine (e.g. MATLAB with the **format long** turned on). This yields:

$$\mathbf{x}\mathbf{y}^T = \begin{bmatrix} 0.617610 & -0.389130 & -0.278460 \\ 1.14180 & -0.719400 & -0.514800 \\ -0.511734 & 0.322422 & 0.230724 \end{bmatrix}$$

b. Simulate the calculation of $\mathbf{x}\mathbf{y}^T$ on a three-digit, decimal based computer which rounds all calculations, including intermediate ones.

solution:

Since this is an outer product, there are no additions or subtractions, and the only intermediate results are the products shown in part a. So all we need to do is round that to three significant digits. This produces:

$$fl(\mathbf{x}\mathbf{y}^T) = \begin{bmatrix} 0.618 & -0.389 & -0.278 \\ 1.14 & -0.719 & -0.515 \\ -0.512 & 0.322 & 0.231 \end{bmatrix}$$

c. Simulate the result of applying Gaussian elimination to this system in part b., on the same notional computer you used in part b. Based on your calculations, what would the theoretical rank be of the matrix?

solution:

From part b. we have that, in a three-digit machine,

$$\mathbf{A} = fl(\mathbf{x}\mathbf{y}^T) = \begin{bmatrix} 0.618 & -0.389 & -0.278 \\ 1.14 & -0.719 & -0.515 \\ -0.512 & 0.322 & 0.231 \end{bmatrix}$$

solution:

Proceeding then with Gaussian elimination we have as the respective multipliers (in a three digit machine)

$$fl(l_{21}) = fl\left(\frac{1.14}{0.618}\right) = fl(1.84466...) = 1.84$$

and

$$fl(l_{31}) = fl\left(\frac{-0.512}{0.618}\right) = fl(-0.828478964...) = -.828$$

Thus, for example, we will, using three-digit rounding arithmetic, replace the element in the second row and column with

$$\begin{aligned} a_{22} - (1.84)a_{12} &= (-0.719) - \overbrace{(1.84)(-0.389)}^{=-0.71576} \\ &= (-0.719) - (-0.716) = -.00300 \end{aligned}$$

Proceeding with the rest of the elimination yields:

$$\begin{array}{l} R_2 - (1.84)R_1 \\ R_3 - (-.828)R_1 \end{array} \quad \left[\begin{array}{ccc} 0.618 & -0.389 & -0.278 \\ 0 & -0.00300 & -0.00300 \\ 0 & 0 & 0.00100 \end{array} \right]$$

(Note the trailing zeros are significant, but the leading ones are **not**! Also note that no further elimination is required, since we are already in echelon form!) Theoretically, \mathbf{A} was a rank three matrix, since it has three non-zero pivots, and therefore three pivot (linearly independent) columns. (Although, obviously, two of the pivots are small, i.e. on the order of (three-digit) machine precision, and therefore, at least in some sense, we expect this will be “close to” a rank one matrix. Alternatively, \mathbf{A} will likely be nearly-singular, or extremely ill-conditioned, at least in a three-digit machine.)

- d. What do your results in parts b. and c. say about whether this calculation is
- (1.) Accurate (Forward Stable)
 - (2.) Backward Stable

solution:

Direct computation will show that

$$\mathbf{x} \mathbf{y}^T - fl(\mathbf{x} \mathbf{y}^T) = \begin{bmatrix} -0.000390 & -0.000130 & -0.000460 \\ 0.001800 & -0.000400 & 0.000200 \\ 0.000266 & 0.000422 & -0.000276 \end{bmatrix}$$

Therefore, using the infinity norm, we have

$$\|\mathbf{x} \mathbf{y}^T - fl(\mathbf{x} \mathbf{y}^T)\|_{\infty} = 0.00240 \quad \text{and} \quad \|\mathbf{x} \mathbf{y}^T\|_{\infty} = 2.3740$$

where, in this instance, both results come from the sum of the magnitudes of the elements in the second row. Therefore,

$$\frac{\|\mathbf{x} \mathbf{y}^T - fl(\mathbf{x} \mathbf{y}^T)\|_{\infty}}{\|\mathbf{x} \mathbf{y}^T\|_{\infty}} = \frac{0.00240}{2.3740} = 0.001010\dots$$

an agreement to well within machine precision on a three digit machine. Therefore the result appears to be accurate, i.e. the computation (algorithm) appears forward stable.

However, as we have already observed, $fl(\mathbf{x} \mathbf{y}^T)$ is a rank three matrix. Since, for any $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$, the product $\mathbf{x} \mathbf{y}^T$ is, in exact arithmetic, of rank one, then there can be **no** $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ such that

$$fl(\mathbf{x} \mathbf{y}^T) = \tilde{\mathbf{x}} \tilde{\mathbf{y}}^T$$

Therefore, by definition, the algorithm is **not** backward stable.